PRIORITY CLAIM

This application claims the benefit of U.S. Provisional Application No. 60/162,412 entitled "Configuring Processing Relationships Among Entities Of An Organization," filed October 29, 1999.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to computer software programs, methods, systems and databases used in Financial Service Organizations. More particularly, the present invention relates to a system and method for configuring processing relationships among entities of a Financial Service Organization (FSO) for use in FSO transaction processing.

2. Description of the Related Art

FSOs such as banks, credit unions, insurance companies, mutual fund companies, credit card companies, brokerage houses, etc., use FSO computer systems to process business transactions. The FSO production systems, which are often referred to as FSO computer systems or FSO systems, are capable of executing application software programs. The application programs may enable FSOs to offer products and services to their clients. The FSO systems may include one or more databases for storing data. The databases may include, for example, groups of data such as the master files of customer account information, transaction-related data such as customer credit card purchase transactions, processing data such as the processing parameters used in processing transactions, and history data such as log files of daily activities for batch processing.

FSO systems often utilize hardcoded software to process FSO transactions. Changes in the business environment often result in corresponding changes to the processing relationship among various entities of a Financial Service Organization (FSO). For example, new banks being acquired or new branch locations being opened often add to the processing structure of an FSO. FSO systems which utilize hardcoded software may be more difficult to adapt to the changing processing structure.

The following is hereby incorporated by reference: Object Oriented mail server framework mechanism (U.S. Patent No. US06081832), Object Oriented framework mechanism for order processing including predefined extensible classes for defining an order processing environment (U.S. Patent No. US06049665), Object Oriented framework mechanism for determining configuration relations (U.S. Patent No. US05937189), Object Oriented mail server framework mechanism (U.S. Patent No. US05768505).

SUMMARY OF THE INVENTION

An improved method, system and carrier medium may be used to configure a Financial Service Organization (FSO) production system. Such a production system typically gathers business data (including transactional data), stores the data, sorts the data, and collates the data into FSO reports used, for example, by various entities of the FSO.

In one embodiment, a multilevel business structure, which may represent the processing relationship between various entities of the FSO, may be configured. A processing relationship configuration program may be used to configure, and subsequently modify, a processing relationship structure. A multilevel node structure may be defined to correspond to the processing relationship structure within an FSO. In one embodiment, one or more rows and one or more columns may represent the multilevel node structure. In one embodiment, a node may be created and uniquely defined to represent an FSO physical entity and/or an FSO function. In one embodiment, examples of an FSO physical entity may be a bank, a branch office, a department, etc. An FSO function, in one embodiment, may be an issuance of a credit card, for example.

In one embodiment, the user may construct a processing relationship structure by selecting a required processing relationship object from one or more objects represented on a display screen. The user may specify the values associated with the selected processing relationship object which may include level number, object name, object identifier, etc. More than one processing relationship object of the same type may be created e.g., multiple bank objects may be created as instances of a bank object. In one embodiment, a root level structure may include only one node. Nodes beneath a node may be referred to as descendents of the node. A node number may uniquely identify a node object in the processing relationship structure. Each newly created node in the functional relationship structure may be assigned a node identifier. In one embodiment, a user may assign a node identifier and the processing relationship configuration program may assign a node number. In one embodiment, the node identifier may be unique. By defining each of the nodes at each of the levels of the processing relationship structure the user may complete the configuration process. In one embodiment, an FSO database may be used to store the processing relationship structure information.

In one embodiment, any node and its relationship with other nodes may be edited to reflect current business conditions by using the edit processing function included in the processing relationship configuration program. In one embodiment, editing may include node operations such as insert, delete, change or expand. In one embodiment, FSO software, such as a program to generate reports, may use the processing relationship structure information to reflect current business conditions.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1a is a block diagram illustrating one embodiment of an FSO computer system for configuring processing relationships;

Figure 1b illustrates one embodiment of an FSO computer system integrated into a networked system for processing FSO business data;

Figures 2a-2e illustrate various embodiments of configuring a processing relationship structure that may be modeled after an FSO business organization structure;

Figure 2f is an example of one embodiment of a multilevel business processing relationship to be modeled in an FSO business system;

Figure 3 illustrates one embodiment of an interactive computer display screen for configuring processing relationships, with a first level of objects representing entities in the FSO displayed;

Figure 4 illustrates one embodiment of an interactive computer display screen for configuring processing relationships, with a first and second level of objects displayed;

Figure 5 illustrates one embodiment of an interactive computer display screen for configuring processing relationships, with a first, second, and third level of objects displayed;

Figure 6 illustrates one embodiment of an interactive computer display screen for configuring processing relationships, with five levels of objects displayed;

Figure 7 illustrates one embodiment of an interactive computer display screen for creating instances of a first object in a processing relationships structure;

Figure 8 illustrates one embodiment of an interactive computer display screen for creating instances of a second object in a processing relationships structure;

Figure 9 is an example of one embodiment of a computer model of the a multilevel business processing relationship illustrated in Figure 2, wherein values have been assigned to the objects in the processing relationship;

Figure 10a is a high-level flow chart illustrating one embodiment of a method of configuring processing relationships for use in configuring reports in an FSO system;

Figure 10b is a mid-level flow chart illustrating one embodiment of a method of configuring processing relationships for use in configuring reports in an FSO system;

Figure 10c is a detailed flow chart illustrating one embodiment of a method of defining processing relationship objects and arranging them in a processing relationship model;

Figure 10d is a detailed flow chart illustrating one embodiment of a method of defining instances of the processing relationship objects defined in Figure 10c; and

Figure 11 is an embodiment of a database table in the FSO system that may be used to store node identifier permutations and node numbers.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION OF THE PREFERED EMBODIMENTS

The term "computer system" as used herein generally describes the hardware and software components that in combination allow the execution of computer programs. The computer programs may be implemented in software, hardware, or a combination of software and hardware. A computer system's hardware generally includes a processor, memory media, and Input/Output (I/O) devices. As used herein, the term "processor" generally describes the logic circuitry that responds to and processes the basic instructions that operate a computer system. The term "memory medium" includes an installation medium, e.g., a CD-ROM, or floppy disks; a volatile computer system memory such as DRAM, SRAM, EDO RAM, Rambus RAM, etc.; or a non-volatile memory such as optical storage or a magnetic medium, e.g., a hard drive. The term "memory" is used synonymously with "memory medium" herein. The memory medium may include other types of memory or combinations thereof. In addition, the memory medium may be located in a first computer in which the programs are executed, or may be located in a second computer that connects to the first computer (e.g., over a network). In the latter instance, the second computer may provide the program instructions to the first computer for execution. In addition, the computer system may take various forms, including a personal computer system, mainframe computer system, workstation, network appliance, Internet appliance, personal digital assistant (PDA), television system or other device. In general, the term "computer system" can be broadly defined to encompass any device having a processor that executes instructions from a memory medium.

The memory medium preferably stores a software program or programs for configuring the FSO system software programs and databases in an FSO system, and for processing FSO transactions in the FSO system, as described herein. The software program(s) may be implemented in any of various ways, including procedure-based techniques, component-based techniques, and/or object-oriented techniques, among others. For example, the software program(s) may be implemented using ActiveX controls, C, C++ objects, JavaBeans, Microsoft Foundation Classes (MFC), or other technologies or methodologies, as desired. A CPU, such as the host CPU, executing code and data from the memory medium includes a way to create and execute the software program or programs according to the methods, flowcharts, and/or block diagrams described below.

A computer system's software generally includes at least one operating system, which is typically a specialized software program that manages and provides services to other software programs on the computer system. Examples of operating systems may include, but are not limited to: Windows NT available from Microsoft Corporation, and the MVS and OS/390 operating systems available from IBM. Software may also include one or more programs to perform various tasks on the computer system and various forms of data to be used by the operating system or other programs on the computer system. The data may include but are not limited to databases, text files, and graphics files. A computer system's software generally is stored in non-volatile memory or on an installation medium. A program may be copied into a volatile memory when running on the computer system. Data may be read into volatile memory as the data is required by a program.

The following sixteen paragraphs introduce various terminologies, definitions, abbreviations, etc., as used herein to describe various embodiments.

As used herein, a Financial Service Organization (FSO) is a business organization that provides financial services to customers and client organizations. As used herein, the term customer generally refers to an individual, and client organization generally refers to other businesses, including retail businesses and other FSOs. Services provided to customers and client organizations include credit products, such as loans and credit cards. An FSO may also provide services to client organizations such as credit card transaction processing. Examples of FSOs include, but are not limited to, banks, credit unions, insurance companies, mutual fund companies, credit card companies and brokerage houses. An FSO that issues credit cards and processes credit card transactions may be referred to as a credit card institution. An FSO may include one or more organizational units. Examples of organizational units include, but are not limited to, main offices, divisions, regional offices, and branch offices.

As used herein, an FSO transaction may be defined as an occurrence of a service provided to a customer or client organization. Examples of FSO transactions include, but are not limited to, financial transactions such as deposits, withdrawals, loan application servicing, and credit card application servicing. FSO transactions may also include services related to financial products such as loans and credit cards previously issued to FSO customers and client organizations. These services may include processing of credit card purchases and collection of payments.

An FSO system may include a data dictionary. A data dictionary may be defined as a collection of descriptions of data items in the database. A description of a data item in a database may be called a data element. A data item may be referred to as a data element value. A data element in the data dictionary may describe attributes of a data element value in the database. Examples of attributes of data element values include, but are not limited to: location in the database, size, and data type. For example, an FSO system data dictionary may describe the data elements involved in credit card processing. The data dictionary may describe each of the data elements in the database for credit card processing. A collection of data may include data elements defined in the data dictionary. Examples of a collection of data, may include, but not be limited to, customer account master files and daily transaction-related data. Examples of data elements in an FSO data dictionary include, but are not limited to: customer name, credit card type, and card issuer.

As used herein, a key is one or more data elements in a database record or group of records that may be used to identify the record or group of records. For example, a record for storing information about an individual may have a name data element. The name data element may be used as a key to identify a particular individual's record in the database. A key value is an instance of a key in the database. In the example above, an example of a key value for a name data element used as a key might be "John Smith." In some examples, not all data elements in a database may be available for use in keys. Data elements that are available for use in keys may be referred to as key elements.

The format of a key may be stored in a key definition. A key definition may include one or more key elements that in combination make the key. During configuration of an FSO system, key definitions may be used in creating key values for records or groups of records in the database. During processing, key definitions may be used by the FSO system to create key values and to read key values stored in the database. During the processing of a transaction, the FSO system may create a key value from transaction-related data using a key definition to extract data element values from the transaction-related data, and may compare the key value to key values stored in the database while searching for a matching key value. A key value created during processing from a key definition and transaction-related data may be referred to as a processing key value.

As used herein, the term "break key" has a different meaning than the term "key" described above. A break key may be defined as a field in a record that may be used as a sort field and/or as a collating field for the record. For example, a set of records may have a field A designated as a break key. Each record's break key field may be set to a break key value. When the records are sorted by a sort process, the sort process may sort the records on the break key field. The sort may be done in ascending or descending order. A break key field may be used in collating the records after a sort. For example, a first group of records with a first break key value may be written to a first file, and a second group of records with a second break key value may be written to a second file.

The FSO system database may include processing parameters used in processing transactions. Processing parameters may be used to apply business logic to the transactions during processing. An example of a transaction processed in an FSO system is a credit card purchase transaction. An example of a processing parameter is a credit card purchase transaction price that may be charged to a client of a credit card institution for the processing of a credit card purchase transaction. An instance of a processing parameter in the database may be referred to as a processing parameter value. For example, an instance of a credit card purchase transaction price might be "$1.50." In some cases, a processing parameter value may include more than one data value. For example, a matrix of data values used in transformation functions on tables of data may be stored as a processing parameter value.

An FSO transaction processing software program may use one or more processing parameters during the processing of a transaction. A processing parameter may have a different processing parameter value for different transactions. The software program may examine the values of one or more data elements in the transaction data and master files to determine the processing parameter value for the transaction. A combination of data elements used to determine the processing parameter value may be referred to as the key definition for the processing parameter. The combination of data element values constructed from the key definition may be referred to as a key value. For example, a software program for processing credit card transactions for a credit card institution may use the credit card issuer and card type to determine what transaction price to charge a client of the credit card institution for processing a credit card transaction. The key definition in this example includes the credit card issuer data element and card type data element, and the key value is constructed from the values for the credit card issuer data element and card type data element read from the credit card transaction data or from a master file associated with the transaction.

In one embodiment, processing parameter values and the key values used to identify the processing parameter values may be stored in tables in the database. The tables in the database that store the processing parameter values and key values may be referred to as Process Control Data (PCD) tables or processing parameter tables. In one embodiment, there may be one PCD table for each processing parameter in the FSO system.

Processing parameters are one example of parameters that may be stored in PCD tables and located using key definitions as described herein. Examples of other types of parameters that may be stored in PCD tables are default parameters and definition parameters. Default parameters may be used to fill in default information in records in the database when they are created. For example, when a new customer account is created, one or more fields in the customer account master file may be filled with default parameter values. Default parameter values may be retrieved from PCD tables using key values constructed from the PCD key definitions and data element values from the customer account master file. Definition parameters are text or numeric values that are located using key values as codes. An example is a text error message that may be looked up using a numeric error code as a key value.

During processing, an FSO transaction may be stored as a record or file in the FSO system. In one embodiment, the FSO transaction may be stored in the FSO system database. A portion of the FSO transaction record may be read into system memory during processing. An FSO transaction record may include one or more data elements. The data elements included in an FSO transaction record may be defined in the data dictionary. The data elements in the transaction record may describe the various attributes of the transaction. For example, the data elements in a credit card transaction record may include items such as the customer's name, account numbers, credit card type, card issuer, date of the transaction, and the business at which the transaction originated.

An example of an FSO that may use an FSO computer system as described herein is a credit card institution. A credit card institution may issue credit cards to customers and client institutions of the FSO. The credit card institution may also issue credit cards on behalf of client businesses such as department stores. The credit card institution may also acquire and process credit card transactions from customers and client businesses such as department stores. For example, a credit card institution may issue its own credit card. Continuing the example, the credit card institution may also have client department stores. The credit card institution may issue a credit card under a department store's name, and may collect and process all credit card transactions for the department store. The credit card institution may charge a fee for each transaction processed. Some of the credit card transactions collected by the credit card institution may be transactions for credit cards not issued by the credit card institution. These transactions may be forwarded to the FSO that issued the card. In turn, other FSOs may forward credit card transactions to the credit card institution. Transactions for credit cards issued by the credit card institution may be processed by the credit card institution.

In the above example, the fee charged for each transaction, also called the merchant transaction price, is an example of a processing parameter for an FSO system in a credit card institution. One embodiment of an FSO system database in a credit card institution may include a merchant transaction pricing PCD table. The merchant transaction pricing PCD table may include one or more merchant transaction pricing values. Each merchant transaction pricing value may be associated with one unique key value in the table. The key values in the PCD table may be constructed using a key definition. Each processing parameter in the FSO system, and thus each PCD table, may be associated with a key definition. In one embodiment, the FSO system database may include a key definition table for storing key definitions in the FSO system.

A key definition may include one or more data elements from the data dictionary. As an example, the merchant transaction pricing parameter described above may have a key definition that includes one or more data elements. Examples of data elements that may be included as fields in the merchant transaction pricing parameter key definition include card issuer, card type, on us/not on us, and transaction type. A card issuer may be the brand of card, for example, VISA, MasterCard, Discovery, etc. Examples of card types may include, but are not limited to: "gold" and "platinum" cards issued by some card issuers. On us/not on us refers to whether the FSO processing the transaction also issued the credit card. "On us" may mean that the FSO did issue the card. "Not on us" may mean that another FSO issued the card, and thus the transaction may be forwarded to the other FSO for processing. The term "transaction type" may refer to the way the transaction was entered; examples of transaction types may include, but are not limited to: manual, electronic, and telephone transactions. A manual credit card transaction may be a credit card transaction that is entered by hand and imprinted with a credit card imprint machine. An electronic transaction may be a credit card transaction where the magnetic strip on a credit card is read electronically. A telephone transaction may be a credit card transaction performed by telephone call.

The organizational and/or functional structure of an FSO may be represented in a FSO computer system by a processing relationship structure. A processing relationship structure may be defined as a computer representation of the entities in the FSO and of the relationships among the entities, wherein the computer representation is useable by software applications to process FSO business data based upon the organizational and/or functional structure of the organization. In one embodiment, the processing relationship structure may be stored in a database on the FSO computer system. In one embodiment, the processing relationship structure may be configured by a user of the FSO system at configuration of the FSO system or during FSO system runtime. Configuration of the FSO system may occur at the time the FSO system software programs and databases are initially installed and set up for processing FSO transactions. Configuration of the FSO system may also occur after the initial configuration performed during the installation of the FSO system. A configuration of the FSO system that occurs after the initial configuration may be called a reconfiguration of the FSO system.

The data dictionary may include one or more processing relationship data elements. One or more processing relationship data elements may be included as key fields in a key definition. In one embodiment, a processing relationship data element may be a Processing Relationship Node Number data element, and the Processing Relationship Node Number data element may be included in a key definition. Thus, a processing parameter value may be located in a PCD table using processing relationship data elements such as the Processing Relationship Node Number data element. This allows processing parameter values to be located for transactions based upon the ownership of the account for which the transaction was generated.

Figure 1a - A block diagram illustrating one embodiment of an FSO computer system for configuring processing relationships

In Figure 1a, an embodiment of an FSO business transaction processing system 10 may include a computer system 20, a display screen 40 connected to the computer system, and one or more databases 52 residing on external storage. Computer system 20 may include memory 30 configured to store computer programs for execution on computer system 20, and a central processing unit (not shown) configured to execute instructions of computer programs residing on computer system 20. Processing relationships configuration program 50 may be stored in memory 20. System 10 may also include a business transaction processing program (not shown). In one embodiment, processing relationships configuration program 50 may be integrated in the business transaction processing program, so that configuring processing relationships may be viewed as a function within the business transaction processing program. System 10 may also include one or more input devices 42 such as a keyboard for entering data and commands into program 50 and one or more cursor control devices 44 such as a mouse.

Figure 1b - One embodiment of an FSO computer system integrated into a networked system for processing FSO business data

Figure 1b illustrates one embodiment of a networked system configured for processing FSO business data. Network 60 may be a local area network or wide area network, and may include communications links including, but not limited to, Ethernet, token ring, internet, satellite and modem. An FSO computer system 10 as illustrated in Figure 1a may be connected to network 60. One or more user workstations 80 may be connected to network 60 and may be used by users of the FSO computer system to enter and modify data, initiate data processing tasks, and monitor the processing of data in the FSO computer system. One or more printers 90 for outputting hardcopy reports on FSO system data may also be connected to network 60. One or more other FSO computer systems 70 may also be connected to network 60. In one embodiment, one or more computer systems 75 of client businesses of the FSO may also be connected to network 60. Client businesses of the FSO may forward business transactions to the FSO computer system for processing. In one embodiment, computer systems 75 may include computer systems belonging to one or more entities within the FSO, such as branches, regional offices, banks, departments, etc.

Figures 2a-2f - Various embodiments of configuring a processing relationship structure that may be modeled after an FSO business organization structure

A Financial Service Organization (FSO) is a business organization that provides financial products and/or services to customers and/or client organizations. An FSO may include one or more organizational units. Examples of organizational units include, but are not limited to, an entity, a business unit, a subsidiary, a division, a functional unit, a headquarters, an operating unit, a profit center, a regional office, and a branch office.

Figure 2a illustrates an example of an FSO business organization according to one embodiment. For example, the FSO business organization may be a global bank 2250. The FSO business units may be represented in a chart or a similar graphical form to illustrate the attributes of an FSO organization such as, but not limited to, the reporting relationship between various FSO entities, the reporting structure, the number of hierarchical levels between the highest level entity and the lowest level entity, and the number of direct reports for an FSO entity. Each FSO entity may be represented as a node or a block on an FSO organizational chart. For example, global bank is represented as node 2250, the business unit for Americas by node 2252, the business unit for Europe, Middle East and Africa by node 2254. Each node may have a parent node and one or more children nodes. For example, USA business unit 2256 has a parent node Americas 2252 and has two children nodes, region AUE 2260 and region AUW 2258. Each node may be identified uniquely with a node number and/or a name. The FSO organizational chart may include multiple levels 2266 in the hierarchical relationship. A node without a parent may be described as a root node or a level zero node. A root node may include the entire FSO organization. The global bank node 2250 may be described as a root node. The FSO organizational chart may be updated, in real-time, as new FSO entities are introduced or removed by adding or deleting a node corresponding to the FSO entity. The FSO organizational chart may thus graphically represent the current, real-world state of the FSO organization.

In one embodiment, an FSO user may create a similar or identical processing relationship structure modeled after the FSO business organization. In one embodiment, an FSO user may use a processing relationship configuration software program to configure or define the processing relationships between various FSO entities which represent the FSO business organization. In one embodiment, an FSO user may configure a node in the processing relationship structure to provide the same or similar functionality provided by the real-world FSO entity. In one embodiment, there may be a one-to-one correspondence between a node included in the FSO business organization chart and a node included in the processing relationship structure.

In one embodiment, the processing relationship structure 2276 may be based on object-oriented technology. Each node in the processing relationship structure 2276 may be represented by a software object which may be defined by the methods and properties associated with the object. For example, in one embodiment, a node may be represented by a bank object. The bank object may include properties such as, but not limited to, bank locations, ATM locations, types of customer accounts, types of loans. The bank object may include methods such as, but not limited to, add_new_account, add_new_location, delete_current_loan. In one embodiment, an FSO user may create various classes of objects such as a class of bank objects. A user may create an instance of the class to create, for example, a new global bank. The new global bank object may inherit all of the properties and methods associated with the class of bank objects.

In one embodiment, the processing relationship structure 2276 may be represented graphically on a display screen 2270, as illustrated in Figure 2b. A user of an FSO may modify or edit the processing relationship structure 2276 by adding or deleting a node, e.g. the object associated with the node. In one embodiment, the node or object may be represented on a display screen 2270 as an icon or a symbol. In one embodiment, a group of objects, each represented as an icon, may be displayed as palette of objects 2274 on a display screen. In one embodiment, the user may use drag-and-drop techniques to add a new object selectable from a palette of objects 2274 to the processing relationship structure. For example, the FSO user may position a cursor 2268 on a node object 2274 and use a drag-and-drop method 2272 to place the selected object 2274 on the processing relationship structure. The FSO user may then configure the node, e.g., the object, by using and/or defining the properties and methods associated with that node.

In one embodiment, the processing relationship structure may be based on traditional programming and traditional database technology. Programming in the C language may be an example of traditional programming. Examples of traditional database technologies may include, but not be limited to, hierarchical, proprietary, relational, flat file. Each node in the processing relationship structure may be represented, in one embodiment, by a table in a relational database. A node may be defined by the rows and columns associated with the table. For example, in one embodiment, a bank table may represent a node. The bank table may include attributes such as, but not limited to, a node identifier, a level number, a sequence number, a bank location identifier, an ATM location description, a customer account number, a type of loan. Access to the bank table may include identifying required keys such as, but not limited to, a transaction identifier, an account number, an FSO user identifier. In one embodiment, the processing relationship structure may be represented by text on a display screen 150, as illustrated in Figure 2c-d. The parent/child or a precedent/descendent relationship may be defined in one embodiment by defining a previous node identifier and a next node identifier. An FSO user may modify or edit the processing relationship structure by adding or deleting a row in a table associated with the node being edited. The columns 152-162 shown in Figures 2c-2e are further described with reference to Figure 3. The FSO user may add the root level node 2250 in Figure 2c. In one embodiment, the FSO user may add a first row to a global bank table. The user may configure the processing relationship structure by entering values for attributes such as, but not limited to, a node identifier, a level number, a sequence number. In Figure 2d, the FSO user may insert a row to add node 2252 for Americas. The user may configure the new node by entering values for attributes such as, but not limited to, a node identifier, a level number, a sequence number. In Figure 2e, the FSO user may insert a row to add node 2254 for Europe, Middle East and Africa. The process may be repeated for all of the remaining nodes included in the global bank business organization chart in Figure 2a. The FSO user may perform a modification to the processing relationship structure, e.g., may reconfigure based on changes in the real world.

In one embodiment, an expert system may perform all the functions of an FSO user. An expert system may be programmed to duplicate or re-create all of the functions performed by the FSO user. For example, an expert system may graphically configure and/or modify the processing relationship structure.

In one embodiment, it may be possible to make the processing relationship structure substantially identical to the FSO business organization. By using the same objects and/or tables, the FSO user may eliminate the need to map real-life FSO entities with corresponding objects which replicate the properties and/methods associated with the real-life FSO entities. Thus, the FSO user may automatically create and/or update a processing relationship structure when the user creates and/or modifies the FSO business organization structure. In one embodiment, the user may be able to create separate processing relationship objects from the FSO entity objects such that the processing relationship objects may be able to automatically mirror or track their corresponding master FSO objects.

The processing relationship structure may be used by FSO application software programs to process FSO transactions. Examples of application software which may utilize the processing relationship structure, may include, but are not limited to, a report generation program, a credit card transaction processing program, a billing program, a monthly account reconciliation summary program. In one embodiment, changes made to the node associated objects and/or tables may have little or no effect on the application software program source code. For example, in Figure 2a the global bank may reorganize its visa account business unit 2262 such that the visa unit now falls under region AUW instead of region AUE. This change may have little or no impact on the report generation program source code for the visa account business unit 2262 since all the objects and/or tables associated with the visa account node, i.e., the owner of the data, may be automatically updated when the FSO user makes changes to the processing relationship structure. The application programs may reference the current properties and/or attributes of the node objects and/or tables to process FSO transactions.

Figure 2f through Figure 9 further illustrate various embodiments of configuring a processing relationship structure by starting with a representative FSO organization structure in Figure 2f and ending with a corresponding processing relationship structure in Figure 9. Figures10a-10d include various flow charts illustrating one embodiment of a method of configuring processing relationships for use in an FSO application software program, such as a report program.

Figure 2f - An example of one embodiment of a multilevel business processing relationship to be modeled in an FSO business system

Figure 2f graphically illustrates one example of a multilevel business processing relationship that may be modeled in an FSO business system using a processing relationships configuration program according to one embodiment. An FSO user or any other person or persons familiar with the FSO organization may create a graphical diagram similar to Figure 2f to reflect the FSO business organization.

In this example, six levels are shown (levels 0-5). Level 0 may be called the root level of the processing relationship structure. Only one node appears at level 0. Node 250 at level 0 represents the root level of the FSO processing relationship structure. All other nodes in the structure are beneath node 250. Nodes beneath a node may be called descendents of the node. At level 1, one or more nodes may appear. In this example, an issuer node 252 and an acquirer node 254 are shown.

In the processing relationship structure, some nodes may represent physical entities in the FSO, and others may represent functional areas. A physical entity is an organizational unit that has a physical presence or manifestation, such as a bank branch office, regional office, or credit card line. A node representing a functional area is used to organize one or more other nodes into a sub-processing relationship group in the FSO processing relationship based upon some function of the FSO. Examples of functional areas include issuer, acquirer, and non-risk. The issuer function may be described as the function of issuing credit cards or other credit instruments to customers of the FSO. The acquirer function may be described as the function of acquiring payments from users of credit cards and other credit instruments on behalf of the FSO and client organizations of the FSO. Non-risk is a functional area that may be used to group nodes dealing with non-risk (or very low risk) instruments such as some types of bonds and secured loans.

Three nodes are shown as descendents of issuer node 252 at level 2. One of the three nodes is Company B 256. Note that Company B is also represented by a node underneath acquirer node 254. In a processing relationship, an entity may appear below more than one functional area. An entity may have more than one function, and the functions of an entity may be represented by separate nodes in the processing relationship.

Company B node 256 has two descendents at level 3, Non-risk node 260 and Bank node 258. A node in the processing relationship tree may represent an object in the processing relationship, such as a bank. During the configuration, a node may be given multiple instances of the object. For example, bank node 258 may represent banks in general at level 3 under Company B. When configured, multiple banks may be created as instances of bank node 258. For example, First Street Bank, Main Street Bank, and Elm Street Bank may be added to the processing relationship structure as instances of Bank node 258. In this example, the three banks report to Company B, but not to each other. Non-risk node 260 has one descendent at level 4, Bank 262. Bank 262 has one descendent at level 5, Branch 264.

By using a processing relationships configuration program and its associated display screens, as described in Figures 3-9, the FSO user may configure the processing relationship structure. At the end of the configuration process, Figure 9 may describe a processing relationship structure, which may be equivalent to the multilevel business processing relationship illustrated in Figure 2f.

<u>Figures 3-8 – Various embodiments of configuring a processing relationship structure using various interactive computer display screens generated by a processing relationship configuration program</u>

Figures 3-8 describe various embodiments of configuring the processing relationship structure, described in Figure 2f, using various interactive computer display screens generated by a processing relationship configuration program.

<u>Figure 3 - One embodiment of an interactive computer display screen for configuring processing relationships, with a first level of objects representing entities in the FSO displayed</u>

Figure 3 illustrates one embodiment of a screen 150 for the user configuration of processing relationships using a processing relationship configuration program in an FSO system. In one embodiment, screen 150 may be presented to a user of a processing relationship configuration program in response to the user selecting an "edit processing relationship" or "create processing relationship" function choice. Function choices may be selectable in a variety of methods, including standard GUI methods such as buttons, menus, text boxes, and mouse highlighting and selection. In one embodiment, the screen 150 may be a graphical user interface (GUI). In another embodiment, the screen 150 may be a textual interface. Screen 150 may include one or more function choices (not shown). For example, screen 150 may include insert node, delete node, edit node, and expand node function choices. Screen 150 may include multiple rows, with each row displaying one node in the processing relationships structure, and multiple columns, with each column displaying one property of the nodes displayed in the rows. The data in the rows and columns may be modifiable by a user of the system. New rows may be inserted in screen 150 by selecting an insertion point and selecting an "insert" function choice from screen 150. In this example, two rows have been inserted. The user may then insert data in one or more of the columns to configure the node. Some of the columns may be automatically filled in by the processing relationship configuration program upon creating the new node.

Columns in screen 150 may include a sequence column 152, an element ID constant column 154, a level column 156, an element ID column 158, an abbreviation column 160, and a description column 162. Sequence column 152 may display a sequence number for the rows in screen 150. In one embodiment, the Element ID constant column 154 may identify an attribute of a node that participates in the processing relationship structure. Element ID constant column 154 may be used to enter, display, and edit a textual identifier for the node. Level column 156 may be used to display the level of a node in the processing relationship structure. For example, Figure 2f illustrates level 0 through level 5. Element ID column 158 may be used to enter, display, and edit an alphanumeric database identifier for the node. Abbreviation column 160 may be used to enter, display, and edit a short label for the node. Description column 162 may be used to enter, display, and edit a textual description of the node.

In one embodiment, an FSO user may create a new object by using the drag-and-drop method. Screen 150 may be displayed in response to an FSO user further identifying the values associated with the methods and properties of the newly created object. In one embodiment, the FSO user may identify specific values for a sequence column 152, an element ID constant column 154, a level column 156, an element ID column 158, an abbreviation column 160, and a description column 162 associated with the newly created object.

Figure 4 - One embodiment of an interactive computer display screen for configuring processing relationships, with a first and second level of objects displayed

Figure 4 illustrates one embodiment of a screen 150 for the user configuration of processing relationships using a processing relationship configuration program in an FSO system. This example shows screen 150 from Figure 3 with more rows added. The first row, or node, at level one in the processing relationship structure, is shown highlighted, signifying that it is selected, with three rows, or nodes, added at level two below the selected first node. The new rows were inserted in screen 150 by selecting the first row as the insertion point and selecting an insert function choice from screen 150. The columns in the new rows were then filled in. Some of the columns may be filled in by the user adding the rows, and some may be automatically filled by the processing relationship configuration program. In this example, the sequence column 152 and the level column 156 may be automatically calculated and displayed by the processing relationship configuration program when a new row, or node, is added. The other columns (154, 158, 160, and 162) may be filled in by the user.

Figure 5 - One embodiment of an interactive computer display screen for configuring processing relationships, with a first, second, and third level of objects displayed

Figure 5 illustrates one embodiment of a screen 150 for the user configuration of processing relationships using a processing relationship configuration program in an FSO system. This example shows screen 150 from Figure 4 with more rows added. The third row, or node, at level two in the processing relationship structure, is shown highlighted, signifying that it is selected, with two rows, or nodes, added at level three below the selected first node. The user may continue selecting rows and inserting rows and filling in the columns in the newly added rows until a processing relationship structure, such as that illustrated in Figure 2, has been fully defined.

Figure 6 - One embodiment of an interactive computer display screen for configuring processing relationships, with five levels of objects displayed

Figure 6 illustrates one embodiment of a screen 150 for the user configuration of processing relationships using a processing relationship configuration program in an FSO system. This example shows a processing relationship structure, such as that illustrated in Figure 2f, which has been fully defined. The descendents of a first node in the processing relationship structure may appear directly beneath the node; after the descendents of the first node, a second node on the same level may appear, and then the second node's dependents, and so on. One or more columns may be indented to represent the processing relationship structure's levels. In this example, the description fields are indented to represent the levels.

Figure 7 - One embodiment of an interactive computer display screen for creating instances of a first object in a processing relationships structure

Figure 7 illustrates one embodiment of a screen 170 for the user configuration of instances of nodes, or processing relationship objects, using a processing relationship configuration program in an FSO system. Screen 170 may be invoked by selecting one of the rows in screen 150 shown in Figure 6 and selecting an "expand node" function choice from screen 150. Screen 170 may include one or more function choices (not shown). For example, screen 170 may include insert row, delete row, and edit row function choices. Function choices may be selectable in a variety of methods, including standard GUI methods such as buttons, menus, text boxes, and mouse highlighting and selection. Screen 170 may include one or more rows, with each row displaying one instance of a node in the processing relationships structure, and one or more columns, with each column displaying one node identifier. The data in the rows and columns may be modifiable by a user of the system. New rows may be inserted in screen 170 by selecting an insertion point and selecting an insert function choice from screen 170. In this example, two rows have been inserted.

Each instance of a node in the functional relationship structure may be assigned a node identifier. The node identifier may be unique among other instances of the node. For example, in the processing relationship structure displayed in Figure 6, the user may select the ISSUER node (row 1, level 1) and assign it a node identifier of 10. The user may then select the COMPANYA node (row 2, level 2) and assign it a node identifier of 1. The user may then select the REGION node (row 3, level 3) and assign it a node identifier of 10. The user may then select the BRANCH node (row 4, level 4) and select the "expand node" function choice from screen 150. Screen 170 may be displayed, with ISS column 172, COA column 174, RGN column 176, BRN column 178, and NODE NUMBER column 179. Initially, no rows may be displayed, as no instances of the selected node may have been created. A row may be inserted in screen 170 by selecting an insertion point and selecting an "insert" function choice from screen 170. The node identifiers for the nodes may then be entered by the user. In one embodiment, one or more of the node identifiers may be automatically filled in by the process relationship configuration program upon inserting a new row. The processing relationship configuration program may then assign a unique node number (not to be confused with the node identifier) to the newly created node instance, and display the node number in NODE NUMBER column 179. In one embodiment, the user may not change the node numbers.

The combination of node identifiers assigned to an instance of a node may uniquely locate the node instance in the processing relationship structure. The node number may be used as an abbreviation for the combination of node identifiers. In the example illustrated in Figure 7, two node instances are displayed. The first is defined by node identifier combination (ISS = 10, COA = 1, RGN = 10, BRN = 100), and is assigned the node number 4. The second is defined by node identifier combination (ISS = 10, COA = 1, RGN = 10, BRN = 200), and is assigned the node number 5.

In one embodiment, an FSO user may create a new instance of an existing object by using the create_new_instance method. Screen 170 may be displayed in response to an FSO user further identifying the values associated with the methods and properties of the new instance of the object (e.g. node 5 created as an instance of node 4). In one embodiment, the FSO user may identify specific values for ISS, COA, RGN, BRN which may be associated with the newly created object.

Figure 8 - One embodiment of an interactive computer display screen for creating instances of a second object in a processing relationships structure

Figure 8 illustrates one embodiment of a screen 170 for the user configuration of instances of nodes, or processing relationship objects, using a processing relationship configuration program in an FSO system. Figure 8 is another example of screen 170 as illustrated in Figure 7. Figure 8 illustrates that different columns may be displayed for different nodes. In the example illustrated in Figure 8, two node instances are displayed. The first is defined by node identifier combination (ACQ = 20, COB = 2, BNK = 70, BRN = 700), and is assigned the node number 20. The second is defined by node identifier combination (ACQ = 20, COB = 2, BNK = 70, BRN = 800), and is assigned the node number 21.

Figure 9 - An example of one embodiment of a computer model of a multilevel business processing relationship illustrated in Figure 2f, wherein values have been assigned to the objects in the processing relationship

Figure 9 illustrates one embodiment of a computer model of the multilevel processing relationship structure illustrated in Figure 2f, showing the node identifiers assigned by a user, and the node numbers assigned by the processing relationship configuration program, to the nodes and node instances by the methods outlined in Figures 3 through 8. In the example shown in Figure 9, the user has created instances of all the nodes, and assigned node identifiers to all of the nodes. All node instances have been assigned unique node numbers. Node 250, the root level node, and the only node on level zero, has been assigned a node number of 0. In one embodiment, the root level node may serve only as the root level node for the rest of the nodes in the processing relationship structure, and may not have an instance created. In one embodiment, nodes on level one may be specified as subsystem nodes. One instance of issuer node 252 at level one has been created and assigned a node identifier of 10 by the user, and a node number of 1 by the processing relationship configuration program. One instance of acquirer node 254 at level one has been created and assigned a node identifier of 20 by the user and a node number of 14 by the processing relationship configuration program.

In one embodiment, nodes at level two may be specified as Company nodes. At level two, under Issuer node instance 252, one instance of Company B node 256 has been created and assigned a node identifier of 2 by the user and a node number of 6 by the processing relationship configuration program. Also at level two, under Acquirer node instance 254, one instance of Company B node 270 has been created and assigned a node identifier of 2 by the user and a node number of 18 by the processing relationship configuration program. Node instances 256 and 270 illustrate that one entity in an organization may occur as multiple node instances in a processing relationship structure if the entity performs multiple functions in the processing relationship structure and therefore appears under multiple function node instances.

In one embodiment, nodes at levels three and lower may be specified as owner nodes, with a number attached to the owner tag indicating the level, with owner 1 at level three, owner 2 at level four, etc. At level three, one instance of Non-risk node 260 has been created and assigned a node identifier of 20 by the user and a node number of 7 by the processing relationship configuration program. Also at level three, one instance of Bank node 258 has been created and assigned a node identifier of 30 by the user and a node number of 10 by the processing relationship configuration program. At level four, one instance of Bank node 262 has been created and assigned a node identifier of 399 by the user and a node number of 8 by the processing relationship configuration program. At level five, one instance of Branch node 264 has been created and assigned a node identifier of 500 by the user and a node number of 9 by the processing relationship configuration program.

Branch node instances 266A and 266B illustrate the creation of multiple instances of a node by a user. Branch node instance 266A has been assigned a node identifier of 700 by the user and a node number of 20 by the processing relationship configuration program. Branch node instance 266B has been assigned a node identifier of 700 by the user and a node number of 20 by the processing relationship configuration program.

Accounts in an FSO system may be associated with node instances in the processing relationship structure. In one embodiment, an account may be associated with only one node instance. Node instance information may be stored in the account master files, and also may be attached to transactions, files, database records, database tables, and other FSO data objects associated with the accounts. In some embodiments, an account master file or other FSO data object associated with the account may include the entire node identifier permutation that uniquely identifies the node in the tree. In some embodiments, an account master file or other FSO data object associated with the account may include the node number that uniquely identifies the node instance associated with the account.

In Figure 9, examples of account types include customer accounts 265, which are accounts for individuals who do business with the FSO, and merchant accounts 267, which are accounts for merchant businesses who do business with the FSO. In one embodiment, accounts may be associated with node instances at any level of the processing relationship structure. In some embodiments, accounts may not be associated with node instances at one or more levels of the processing relationship structure.

In one embodiment, a database table in the FSO system may be used to store the node identifier permutations and node numbers for all of the node instances in a processing relationship structure, as illustrated in further detail in Figure 11. The database table may be used in the FSO system as a lookup table to convert between node numbers and node identifier permutations. The node number may serve in the FSO system as an abbreviation of the node identifier permutation, and allows the node instance associated with an account to be identified in FSO data objects using a minimum of space. The node identifier permutation may be used in user interfaces, output files, etc. to identify node instances to the user, as this may be the form a user is most likely to recognize and understand.

Figures10a-10d - Various flow charts illustrating one embodiment of a method of configuring processing relationships for use in various FSO application software programs

Figures10a-10d include various flow charts illustrating one embodiment of a method of configuring processing relationships for use in various FSO application software programs, such as a report program.

Figure 10a - A high-level flow chart illustrating one embodiment of a method of configuring processing relationships for use in configuring reports in an FSO system

Figure 10a is a flowchart illustrating one embodiment of a method of creating a processing relationship structure using the processing relationship configuration program, and of then using the node instances created in the processing relationship structure in an FSO runtime system in report record definitions to generate reports based on the processing relationship structure. In step 400, the processing relationship configuration program is used by a user of the FSO system to configure the processing relationship structure and the nodes and node instances therein. In step 402, the user may add one or more node instances to report record definitions, which may then be used to extract, sort, and/or collate reports based upon the processing relationship structure. Figures 10b-10d expand on the flowchart illustrated in Figure 10a.

Figure 10b - A mid-level flow chart illustrating one embodiment of a method of configuring processing relationships for use in configuring reports in an FSO system

Figure 10b is a flowchart illustrating one embodiment of a method of creating a processing relationship structure using the processing relationship configuration program, and of then using the node instances created in the processing relationship structure in an FSO runtime system in report record definitions to generate reports based on the processing relationship structure. Figure 10b expands on the flowchart illustrated in Figure 10a; steps 400 and 402 have been expanded into several steps.

In step 404, one or more persons may first define the business structure of the FSO to be modeled in a processing relationship structure. After the business structure has been defined, a processing relationship structure may be configured using the processing relationship configuration program. In step 406, a user may define the processing relationship objects, or nodes, at each level of the processing relationship structure. The process of defining the nodes at each level of the structure, and of defining descendents of the nodes at the next level, in effect creates the processing relationship structure. In step 408, the user may then select processing relationship nodes in the processing relationship structure and create instances of the nodes by assigning node identifiers to the nodes. A node identifier may uniquely define a node instance within the processing relationship structure, and may define the relationship of the node instance with node instances above it in the processing relationship structure.

In step 410 and 412, report record definitions may be defined. A report record definition is a data structure that defines the format of report records in an FSO system. A report record may be defined as a data record including break key values and one or more data values. A report record may also include other data such as header data, sequencing information, time stamps, etc. Break key values in a report record may be used to sort the report records by one or more of the break key values, thus ordering the report records in a logical sequence by one or more of the break key fields. The break key values may then be used when generating a report to collate the report by one or more of the break key fields. When generating reports, a collection of data records, such as transaction log records, transaction records, account master files, etc., may be processed, with one or more report records extracted from the collection of data records using the report record definition for the report to be generated. In some embodiments, one report record may be generated for each data record processed.

In step 410, a user may add break keys to a report record definition. Break keys are fields in the report record definition that are used to extract data values from transaction records, account master files, or other data sources, and to assign the extracted data values to break key fields in a report record. The report record definition may also include one or more data field definitions which may be used to extract data values from data elements in transaction records, account master files, or other data sources, and to assign the extracted data values to report data fields in a report record. The data field definitions may be added to the report record definition in step 412.

Processing relationship node identifiers may be used in report record definitions as break key fields. In one embodiment, node numbers may be used as an abbreviation for node identifiers in break key fields. A data record for which a report record is generated for a report may be associated with an account in the processing relationship structure. The data record may include a processing relationship node identifier and/or node number that may be used to link the data record to the account in the processing relationship structure. Using processing relationship node identifiers as break key fields in report record definitions may allow reports to be sorted and collated according to the processing relationship structure. This may allow individual reports to be automatically generated for business entities and FSO functional areas based upon the processing relationship structure. Rollup and summary reports may also be generated. As the report records including node instances are processed into reports, summary data may be compiled from the data fields within the report records. When a change, or break, in the key field values for the node instances is encountered, a summary report for all node instances at that level, for that branch of the processing relationship structure, may be generated. A report may continue this process for the entire processing relationship structure, thus generating summary reports for the entire FSO processing relationship as defined in the structure.

Figure 10c - A detailed flow chart illustrating one embodiment of a method of defining processing relationship objects and arranging them in a processing relationship model

Figure 10c is a flowchart illustrating one embodiment of a method of creating a processing relationship structure using the processing relationship configuration program. Figure 10c is an expansion of step 406 from the flowchart illustrated in Figure 10b. In step 414, a user may initialize a new processing relationship structure by creating a processing relationship object, or node, at the first level. In one embodiment, a GUI in the processing relationship configuration program may provide a method of creating a new processing relationship structure. For example, a "create new processing relationship structure" menu choice may be available to the user. In one embodiment, creating a new processing relationship structure may automatically create a first node at a first level of the processing relationship. In one embodiment, this first node may be a system node that represents the highest level in the FSO processing relationship structure. In one embodiment, all other nodes in the processing relationship structure are created as descendents of the first node. The first node may be assigned a unique node number by the processing relationship configuration program. In one embodiment, the node at the first level may be assigned a node number of 0. In one embodiment, the first level may be level 0.

In one embodiment, nodes created in the processing relationship structure may be displayed on a processing relationship configuration screen by the processing relationship configuration program. In one embodiment, one row may be displayed for each node in the processing relationship structure. The row may include fields for displaying and entering information about the node. Fields in the row may include a sequence field, a element ID constant field, a level field, an element ID field, an abbreviation field, and/or a description field. A sequence field may display a sequence number for the row in the screen. An element ID constant field may be used to enter, display, and edit a textual identifier for the node. A level field may be used to display the level of a node in the processing relationship structure. An element ID field may be used to enter, display, and edit an alphanumeric database identifier for the node. An abbreviation field may be used to enter, display, and edit a short, for example three characters, label for the node. A description field may be used to enter, display, and edit a textual description of the node. The rows in the screen may also include other information about the node not described herein.

In step 416, the user may select the first node created in step 414 on the processing relationship configuration screen in preparation for adding nodes at the second level as descendents of the first node. Selecting a node on the processing relationship configuration screen may be accomplished using one of a variety of methods, including standard GUI methods such as mouse highlighting and selection. The processing relationship configuration screen may include one or more function choices. For example, the screen may include insert node, delete node, edit node, and expand node function choices. Function choices may be selectable in a variety of methods, including standard GUI methods such as buttons, menus, text boxes, and mouse highlighting and selection.

In step 418, after the first processing node is selected, the user may insert a node at the second level as a descendent of the first node. The user may invoke an "insert node" function after selecting the first node. A new row representing the newly created node may be displayed immediately below the first node's row. The user may then insert data in one or more of the fields in the row to configure the new node. Some of the fields in the row may be automatically filled in by the processing relationship configuration program upon creating the new node. In step 420, the user determines if there are more nodes to be inserted at the second level as a descendent of the first node. If so, steps 418 and 420 are repeated. If no more nodes are to be added at the second level as descendents of the first node, the first node may be deselected by the user.

In step 422, the user may determine if nodes are to be added as descendents of any node already created in the processing relationship structure. If more nodes are to be added, the user proceeds to step 416 to select a node for which descendent nodes are to be added. Steps 416-422 may be repeated by the user until all processing relationship nodes in the processing relationship structure have been created and configured.

<u>Figure 10d - A detailed flow chart illustrating one embodiment of a method of defining instances of the processing relationship objects defined in Figure 10c</u>

Figure 10d is a flowchart illustrating one embodiment of a method of creating a processing relationship structure using the processing relationship configuration program. Figure 10d is an expansion of step 408 from the flowchart illustrated in Figure 10b. In step 424, the user may select a node displayed on the processing relationship configuration screen in preparation for adding node instances for the node. In one embodiment, nodes created in the processing relationship structure may be displayed on the processing relationship configuration screen by the processing relationship configuration program. In one embodiment, one row may be displayed for each node in the processing relationship structure. The row may include fields for displaying and entering information about the node. Selecting a node on the processing relationship configuration screen may be accomplished using one of a variety of methods, including standard GUI methods such as mouse highlighting and selection. The processing relationship configuration screen may include one or more function choices. For example, the screen may include insert node, delete node, edit node, and/or expand node function choices. Function choices may be selectable in a variety of methods, including standard GUI methods such as buttons, menus, text boxes, and mouse highlighting and selection.

In step 426, after node is selected, the user may add instances of the node. The user may invoke an "expand node" function after selecting the first node. In response to the "expand node" function being invoked, the processing relationship configuration program may open a node instance configuration screen. The screen may include one or more function choices. For example, the screen may include insert row, delete row, and/or edit row function choices. Function choices may be selectable in a variety of methods, including standard GUI methods such as buttons, menus, text boxes, and mouse highlighting and selection. The screen may include one or more rows, with each row displaying one instance of a node in the processing relationships structure, and one or more columns, with each column displaying one node identifier. The data in the rows and columns may be modifiable by a user of the system. A new instance of the node may be created by inserting a new row in the screen. New rows may be inserted in the screen by selecting an insertion point and selecting an "insert function" choice from the screen.

Initially, no rows may be displayed, as no instances of the expanded node may have been created. A row may be inserted in by selecting an insertion point and selecting an "insert" function choice from the screen. The node identifiers for the nodes may then be entered by the user. In one embodiment, one or more of the node identifiers may be automatically filled in by the process relationship configuration program upon inserting a new row. The processing relationship configuration program may then assign a unique node number (not to be confused with the node identifier) to the newly created node instance, and display the node number in a column of the node instance row. In one embodiment, the user may not change the node number. The combination of node identifiers assigned to an instance of a node uniquely locate the node instance in the processing relationship structure. The node number may be used as an abbreviation for the combination of node identifiers. In one embodiment, the node identifiers assigned to an instance of a node may be read from left to right, with the leftmost being the node identifier of the highest node on the branch of the processing relationship structure the new node instance is on. The rightmost node identifier may be the node identifier of the newly created node, and may be assigned and edited by the user. Nodes between the leftmost node and rightmost node may be node identifiers of the ancestors of the node instance on the branch of the processing relationship structure. In one embodiment, the user may assign and edit the node identifiers of ancestor nodes. Other embodiments of ordering schemes for the node identifiers may be used, such as right-to-left, top-to-bottom (vertical arrangement), etc.

In step 428, the user determines if any more node instances are to be created for the currently selected node. If there are more node instances to be created for the currently selected node, step 426 may be repeated until all the node instances have been created. If there are no more node instances to be created for the currently selected node, the user may determine if node instances remain to be created for any other nodes in the processing relationship structure. If there are more node instances to be created, the user may repeat steps 424-428 until all node instances for all nodes have been created.

Figure 11 – An embodiment of a database table in the FSO system that may be used to store node identifier permutations and node numbers

Figure 11 illustrates an embodiment of a database table in the FSO system that may be used to store the node identifier permutations 342 and node numbers 340 for all of the node instances in a processing relationship structure. The database table may be used in the FSO system as a lookup table to convert between node numbers 340 and node identifier permutations 342. The node number 340 may serve in the FSO system as an abbreviation of the node identifier permutation 342, and allows the node instance associated with an account to be identified in FSO data objects using a minimum of space. The node identifier permutation may be used in user interfaces, output files, etc. to identify node instances to the user, as this may be the form a user is most likely to recognize and understand.

A processing relationship node number data element may be used as a key field in key definitions. The key definition may be used to search for processing parameters in the FSO system database. Since a node number uniquely identifies a node instance in the processing relationship structure, and one or more of the node identifier(s) that are associated with the node number may be changed without changing the node number, key definitions do not have to be modified when one or more node identifier(s) in the processing relationship structure are changed. This may isolate the FSO application software program source code from changes when the processing relationship structure may have changed.

In one embodiment, a processing relationship node number may be logged with a transaction record when the transaction is closed or written to a logfile, wherein the processing relationship node number uniquely identifies a processing relationship node instance in the processing relationship structure. The processing relationship node number in the transaction record may then be used by the system to identify the owner of the transaction record's account in the processing relationship structure.

Various embodiments further include receiving or storing instructions and/or data implemented in accordance with the foregoing description upon a carrier medium. Suitable carrier media include memory media or storage media such as magnetic or optical media, e.g., disk or CD-ROM, as well as signals such as electrical, electromagnetic, or digital signals, conveyed via a communication medium such as networks and/or a wireless link.

Although the system and method of the present invention have been described in connection with several embodiments, the invention is not intended to be limited to the specific forms set forth herein, but on the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the spirit and scope of the invention as defined by the appended claims.